METHOD FOR PATCHING ROM INSTRUCTIONS IN AN ELECTRONIC EMBEDDED SYSTEM INCLUDING AT LEAST A FURTHER MEMORY PORTION

Field of the Invention

[0001] The present invention relates to a method for patching read only memory (ROM) instructions in an electronic embedded system that includes an additional memory portion. The additional memory portion may be a ROM, a RAM, an EEPROM or other non-volatile memory devices. The electronic embedded system may be a CPU based system, for example.

[0002] The electronic embedded system may be incorporated into an electronic device, such as a terminal of a wired or wireless communication system that includes a plurality of nodes structured to receive and transmit signals among the terminals. The embedded electronic device may be a personal digital assistant (PDA) or a mobile telephone including an IC-card with resident applications stored in the ROM portion. The invention further relates to such an electronic device, and to an IC-card used in such a device.

Background of the Invention

[0003] Embedded systems based on a read only memory program are equipped with a non-volatile memory potion, for instance an electrically erasable and programmable read only memory (EEPROM) that is used for customized

data. These embedded systems are incorporated into portable electronic devices such as handhelds, mobile phones, IC-cards, etc. Such devices operate according to a program (software application) masked on the read only memory that defines its functionalities.

[0004] The manufacturers, before making available a product based on the masked ROM device, stores customized data and user data into a non-volatile memory portion. As an example, consider the case of an IC-card for a GSM application. The telecommunication operators or providers require as much space as possible for the manufactures of the IC-card in terms of memory space of the non-volatile memory portion. Such space is used for the user data (i.e., address book, SMS, etc.) as well as for their own customized data from the provider (i.e., special operator functionalities, proprietary authentication algorithms, etc.). Generally speaking, the more non-volatile memory space is available the more attractive the final product is from a marketing point of view.

[0005] Moreover, since there is often the need to change or fix the ROM software functionalities without re-masking, it would be very important to have an efficient extending or patching mechanism for hooking new instructions to the ROM for picking them up from the non-volatile memory portion.

[0006] To extend or patch the ROM functionalities, the masked program needs to be sensitive to the patching mechanisms. According to the prior art, these mechanisms are based on the principle of replacing groups of masked ROM instructions, also known as subroutines, with new set of instructions. In other words, a new subroutine is stored in the additional

non-volatile memory portion (e.g., EEPROM, Flash, etc.). This approach, however, suffers two problems. One problem is that the new subroutine stored in the non-volatile memory portion cannot reuse the calling ROM subroutine. A second problem is that it is not possible to reuse the ROM subroutine since the size of an EEPROM subroutine will be as large as the size of ROM subroutine that needs to be replaced.

[0007] Figure 1 is a schematic and simplified view showing how the common patching mechanism of the prior art works. As may be appreciated, during execution of a ROM subroutine there might be a call to the EEPROM subroutine to be executed in place of the ROM instructions. However, after having executed the EEPROM instructions there is no possibility to reuse the ROM instructions without causing a recursive endless action.

Summary of the Invention

[0008] In view of the foregoing background, an object of the present invention is to provide a mechanism for patching ROM instructions in an electronic embedded system, for instance a CPU based system, including at least one additional memory portion.

[0009] Another object of the present invention is to provide a method for patching ROM instructions based on the criteria that the ROM subroutines may be reusable.

[0010] A further object of the present invention is to allow the call and execution of EEPROM instructions having also the possibility to reuse the calling ROM instructions without incurring in a recursive endless action.

[0011] An embodiment of the invention provides a method for patching or extending ROM instructions in an electronic embedded system, for instance a CPU based system, including a first read only memory portion, an extended memory portion and an additional memory portion. Instructions groups, such as subroutines or software modules, defining the patching functionalities of the system are contained in the read only memory portion while extended instructions are stored in the extended memory portion. The processing of the subroutines or instructions of the first read only memory portion is alternated by a checking phase of the logic value of corresponding flags stored in the additional memory portion indicating the need of executing subroutine instructions in the extended memory portion.

[0012] Another embodiment of the invention provides for an electronic embedded system structured to implement the inventive method, and including at least a memory portion storing logic values of corresponding flags indicating the need of executing subroutine instructions stored in main or extended memory portions.

Brief Description of the Drawings

[0013] The features and advantages of the method and system according to the present invention will become apparent from the following description of an embodiment thereof, given by way of non-limiting examples with reference to the accompanying drawings.

[0014] Figure 1 is a schematic and simplified view showing a common patching mechanism according to a prior art method;

[0015] Figure 2 is a schematic and simplified view showing a patching mechanism based on the method of the present invention; and

[0016] Figure 3 is a scheme about the patching mechanism of the invention showing how the subroutines work together as well as their operational sequence.

Detailed Description of the Preferred Embodiments

[0017] With specific reference to the view of Figure 2, an electronic embedded system, for instance a CPU based system, is schematically shown at 1. According to the invention, the system 1 includes at least a first read only memory (ROM) portion 2, a second extended memory portion 3, and at least an additional memory portion 5. The first memory portion 1 may be an electrically erasable memory portion of the EEPROM or Flash type.

[0018] The second and extended memory portion 3 of the system 1 is a non-volatile memory, for instance an EEPROM, including subroutines, extended instructions and or data. The additional memory portion 5 may be structurally and functionally independent from both the first read only memory portion 2 and the extended memory portion 3, and may be a read/write memory such as a volatile RAM. As an alternative, the additional memory portion 5 may be an EEPROM or another non-volatile memory device.

[0019] The invention makes use of microprocessor instructions groups, called a subroutine or a software module, which defines the patching functionalities, and to the data contained in the memory portions 3 and 5.

[0020] Figure 3 shows a schematic flow chart of the possible evolution of the patching mechanism according

to the present invention. The ROM portion 2 stores ROM instructions and subroutines that are executed according to a predetermined software algorithm.

[0021] In such a configuration, the ROM portion 2 contains many application modules (or subroutines) which are subordinate to a flag or semaphore 4 status stored in the additional memory portion 5.

[0022] The flag indicates to the subroutine whether or not to execute their own ROM instructions or the subroutine instructions of the extended memory portion 3, for instance EEPROM based subroutine instructions. The EEPROM based subroutine instructions may be used to patch or to extend the behaviors masked (unchangeable) in the instructions of the ROM portion 2. Furthermore, the EEPROM based subroutine may reuse the ROM based subroutine without incurring in recursive actions.

[0023] Thus, differently from conventional mechanisms, the present invention provides a patching mechanism based on the application of flags 4 placed in the read/write additional memory portion 5, in which the ROM subroutine controls its own execution. With the help of these flags, the ROM subroutines may know whether and when it is possible to call an EEPROM subroutine without resulting in recursive behaviors.

[0024] Furthermore, it is possible to execute the same calling ROM subroutine from the EEPROM subroutines, thus avoiding the need to reproduce the same ROM instructions even in the EEPROM. In other words, in most of the cases an EEPROM subroutine just needs to contain integrative instructions without replacing the ROM instructions in their entirely to fix the behaviors of the calling ROM subroutine.

[0025] In more detail, the data and the subroutine are structured into the memory portions 2, 3 and 5 as follows. In the read only memory 2, there are subroutines called application code which are required to be patched (or to be extended). They contain some instructions predisposed for the patching mechanism according to which it is possible to call, indirectly, a patch (or extension) resident into non-volatile memory.

[0026] In the extended non-volatile memory portion 3, there are subroutines called application patch code which are required for providing new instructions to the application code. These subroutines may be used to fix improper functioning of the ROM application code.

[0027] In the additional read/write volatile memory portion 5, for each application patch code is present information (called a flag) that keeps the status of its execution. The flag may assume two types of values: free or busy (green or red, respectively).

[0028] To define the invention, an interaction mechanism between all the mentioned software modules is provided. The patching mechanism starts from the software module that requires to be patched (or to be extended), therefore, from the application code (ROM). At this stage, before executing the application code instructions, a checking phase about the presence of the correspondent application patch code is performed.

[0029] When the application patch code is not present, then nothing occurs and the execution proceeds with the instructions of ROM application code. On the contrary, if the application patch code is present, the application code checks for the status of the associated application patch code flag 4, and when the

flag is free, the latter is set to a different logic value, for instance a value corresponding to an information busy or red state.

[0030] As a last step, the application code invokes the corresponding application patch code located in non-volatile memory. The patching mechanism terminates when the flag is reset to free (or green) by the application code, right after the application patch code instructions have been completely executed.

[0031] Everything described below is applicable to any subroutine, software module or group of CPU instructions. The flag is binary information associated to each subroutine that uses the patching mechanism. Therefore, there will be as many flags as the number of patchable subroutines. In any case, the mechanism is generic.

[0032] The ROM subroutine is a generic term referring to instructions resident in a generic read only memory (like ROM). The EEPROM subroutine is a generic term referring to instructions resident in a generic non-volatile memory (like EEPROM).

[0033] With specific reference to the example of Figure 3, the invention may be disclosed as the software module to be patched would be a ROM subroutine. In this respect the initial status of the inventive method starts from the phase indicated with (0). Since the ROM subroutine is conditioned by the status of the flag 4, two scenarios are delineated. One with the flag green and the other with the flag red.

[0034] In more detail, when the ROM subroutine starts (0) with a green flag (initial status), the following actions are taken. The ROM subroutine gets information about the presence of the EEPROM subroutine

(1). If the EEPROM subroutine is not present, then the ROM subroutine proceeds with the normal execution of ROM instructions (9) until the end of subroutine (11).

[0035] In the case of the EEPROM subroutine being present, the following are performed. The ROM subroutine checks the status of the flag. If the flag is red then the ROM subroutine proceeds with the normal execution of ROM instructions (9) until the end of subroutine (10) by passing for the return of the control of the ROM subroutine (7) and finally for resetting the flag to the green status (8).

[0036] For a green flag, the following are performed. The ROM subroutine sets the flag to a red status (3). The ROM subroutine calls the EEPROM subroutine (4). The EEPROM subroutine instructions are executed (5), and the EEPROM subroutine calls the calling ROM subroutine (0). In this case the EEPROM subroutine reuses the ROM instructions. This step could be processed even before EEPROM instructions are executed (5) or both, that is, before and after step (5). The EEPROM subroutine returns the control to ROM subroutine (7), and the ROM subroutine sets the flag to a green status (8), and the ROM subroutine ends its execution (11).

[0037] As may be appreciated by the above disclosure of the invention, a new or different patching of the subroutines of a specific application may be programmed according to the present invention by a straightforward definition of a predetermined number of flags that allow addressing and calling back subroutines of the extended memory portion without altering the main read only memory portion.